33. (Amended) A method for enhancing user control over the look and feel of an application program executable on a computer with the help of an operating system, the application program using at least one external resource file and computer executable instructions, the method comprising the steps of: creating the resource file using a markup language; providing a resource-loader routine for retrieving information from the resource file without compiling the resource file by making a call in the computer executable instructions to the resource-loader to obtain information from the resource file; generating the computer executable instructions independent of the resource file; and executing the instructions in the markup language to provide the requested information to the resource-loader.

34. (Amended) A system for developing an application program for execution on a computer with the help of an operating system, the application program having at least one resource file and computer executable instructions, the system comprising: the resource file containing information stored in accordance with a markup language; an executable file corresponding to the application including computer executable instructions for making a call on the resource loader; a resource-loader routine in the operating system for retrieving information from the resource file without compiling the resource file and in response ~~responsively~~ to a call made by the executable file while executing; and an interpreter for executing the instructions in the markup language to provide the requested information to the resource-loader.

35. (Amended) A resource loader in an operating system for accessing information in a resource file, which is not complied, during execution of an application program and providing the information to the application program wherein the resource loader program comprises: an

addressing mechanism for communicating a request from an application program for specified information from at least one resource file; a markup language handling functionality to retrieve data stored using a markup language wherein the data in the resource file is modifiable by a user while the application is executing and wherein the data corresponds to the information requested by the application program from the resource loader; and a second addressing mechanism for directly or indirectly communicating the retrieved data to the application program from the resource loader.

36. (Amended) A computer readable medium having computer executable instructions for carrying out the steps of a method for developing an application program, the application program having at least one graphical interface, the graphical interface having at least one parameter specified in a resource data-containing file, the method comprising the steps of: creating the resource data-containing file, which is not complied, using a markup language to identify the parameter; using a graphical control locator for identifying the resource data-containing file; parsing text in the resource data-containing file; walking the parsed text to provide the parameter to the graphical interface; and carrying out the instructions in the markup language to implement the graphical interface in accordance with the parameter.

PATENT
Attorney Docket No. 202266

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Group Art Unit: 2122

ANDREW et al. Examiner: Das, C.

Application No. 09/452,421

Filed: December 1, 1999

For: EXTERNAL RESOURCE FILES FOR APPLICATION DEVELOPMENT AND MANAGEMENT

PENDING CLAIMS AFTER AMENDMENTS MADE IN RESPONSE TO OFFICE ACTION DATED DECEMBER 16, 2002

1. A method for developing an application program, the application program having at least one graphical interface, the graphical interface having at least one parameter specified in a resource data-containing file, the method comprising the steps of: creating the resource data-containing file, which is not complied, using a markup language to identify the parameter; using a graphical control locator for identifying the resource data-containing file; parsing text in the resource data-containing file; walking the parsed text to provide the parameter to the graphical interface; and carrying out the instructions in the markup language to implement the graphical interface in accordance with the parameter.

2. The method of claim 1 wherein the graphical interface has a property specifiable by the parameter.

3. The method of claim 1 wherein the step of parsing the text generates a tree.

4. The method of claim 3 wherein the step of walking the parsed text uses a resource identifier to identify a node corresponding to a requested graphical property and the parameter associated with the node.

5. The method of claim 1 wherein the graphical control locator parses the text in the resource data-containing file.

6. The method of claim 1 wherein the graphical control locator walks a tree constructed by the step of parsing the text.

7. The method of claim 1 wherein the graphical control locator is a resource loader.

8. The method of claim 1 wherein the resource data-containing file is a resource file.

9. The method of claim 1 furthermore having the step of generating the resource data-containing file with a resource data editor suitable for generating text in a desired markup language.

10. The method of claim 9 wherein the step of generating includes modifying the resource data-containing file.

11. The method of claim 9 wherein the resource data editor provides a name for the resource data-containing file in accordance with a resource identifier in a predefined format.

12. The method of claim 11 wherein the graphical control locator uses the resource identifier for locating the resource data-containing file.

13. The method of claim 1 wherein the text in the resource data-containing file includes script.

14. The method of claim 13 wherein the step of walking the text in the resource data-containing file includes executing the script.

15. The method of claim 1 furthermore having the step of generating an event due to user input responsive to the graphical interface.

16. The method of claim 15 furthermore having the graphical control locator, responsively to the event, modify the graphical interface.

17. The method of claim 15 furthermore having the step of, responsively to the event, transmitting a message from the graphical control locator to the application.

18. The method of claim 15 furthermore having the step of generating an event due to user input responsive to the graphical interface wherein the graphical interface includes multimedia effects.

19. The method of claim 13 wherein the step of creating the resource file is carried out at runtime.

20. An application developing system for implementing an iterative application development strategy involving at least two groups of workers comprising:

a first group of workers for generating functional code for an application in course of developing the application;

a second group of workers who modify the functional code by modifying an external resource data-containing file, wherein the external resource data-containing file includes markup language whereby obviating compilation of the external resource data-containing file for the execution of the functional code;

a graphical control locator for locating the external resource data-containing file responsively to the functional code; and

an external resource data-containing file parser for identifying a requested parameter stored in the external resource data-containing file.

21. The system of claim 20 wherein a resource loader provides both the services of the graphical control locator and the external resource file markup language parser.

22. The system of claim 20 wherein a resource loader provides the services of a parser of the markup language, wherein the parser can be extended by plugging in additional namespaces into a schema used by the parser for describing the markup language.

23. The system of claim 20 further comprising a resource data editor for modifying the external resource data file by introducing a change in a desired markup language.

24. The system of claim 23 wherein the resource data editor allows editing the resource data-containing file while the functional code is executing.

25. The system of claim 24 wherein the resource data editor creates a copy of the resource data-containing file in response to a request from a user to edit the the resource data-containing file if the functional code invoking the resource data-containing file is executing.

26. The system of claim 20 wherein furthermore at least one of the second group of workers cannot access the functional code directly without authorization while modifying the functional code by introducing changes in the external resource data-containing file, whereby enhancing the security of the functional code.

27. The system of claim 20 wherein furthermore at least one of the first group of workers cannot access the external resource data-containing file directly without authorization while modifying the functional code, whereby enhancing the security of the changes made to the external resource data-containing file by the second group of workers.

28. A method for enhancing security in an application-developing environment, wherein a first worker and a second worker cooperate in developing the application, the method comprising the steps of: creating an external resource data-containing file, which is not compiled, for storing data in a markup language for implementing resources; using a graphical control locator for retrieving information from the resource file in response to a request for a resource; providing the first worker with authority to modify the resource data-containing file and execute the application code; and restricting the first worker from accessing or modifying the application source code.

29. The method of claim 28 wherein the method further includes the step of restricting the second worker from accessing or modifying the resource data-containing file.

30. The method of claim 28 wherein the first worker is a designer.

31. The method of claim 28 wherein the second worker is a developer.

32. The method of claim 28 wherein the step of restricting uses passwords or keys to determine if the first worker should be allowed access to the application source code.

33. A method for enhancing user control over the look and feel of an application program executable on a computer with the help of an operating system, the application program using at least one external resource file and computer executable instructions, the method comprising the steps of: creating the resource file using a markup language; providing a resource-loader routine for retrieving information from the resource file without compiling the resource file by making a call in the computer executable instructions to the resource-loader to obtain information from the resource file; generating the computer executable instructions independent of the resource file; and executing the instructions in the markup language to provide the requested information to the resource-loader.

34. A system for developing an application program for execution on a computer with the help of an operating system, the application program having at least one resource file and computer executable instructions, the system comprising: the resource file containing information stored in accordance with a markup language; an executable file corresponding to the application including computer executable instructions for making a call on the resource loader; a resource-loader routine in the operating system for retrieving information from the resource file without compiling the resource file and in response to a call made by the executable file while executing; and an interpreter for executing the instructions in the markup language to provide the requested information to the resource-loader.

35. A resource loader in an operating system for accessing information in a resource file, which is not complied, during execution of an application program and providing the information to the application program wherein the resource loader program comprises: an

addressing mechanism for communicating a request from an application program for specified information from at least one resource file; a markup language handling functionality to retrieve data stored using a markup language wherein the data in the resource file is modifiable by a user while the application is executing and wherein the data corresponds to the information requested by the application program from the resource loader; and a second addressing mechanism for directly or indirectly communicating the retrieved data to the application program from the resource loader.

36. A computer readable medium having computer executable instructions for carrying out the steps of a method for developing an application program, the application program having at least one graphical interface, the graphical interface having at least one parameter specified in a resource data-containing file, the method comprising the steps of: creating the resource data-containing file, which is not complied, using a markup language to identify the parameter; using a graphical control locator for identifying the resource data-containing file; parsing text in the resource data-containing file; walking the parsed text to provide the parameter to the graphical interface; and carrying out the instructions in the markup language to implement the graphical interface in accordance with the parameter.

37. A computer readable medium having computer executable instructions for carrying out the steps of a method for enhancing security in an application-developing environment, wherein a first worker and a second worker cooperate in developing the application, the method comprising the steps of: creating an external resource data-containing file, which is not compiled, for storing data in a markup language for implementing resources;

using a graphical control locator for retrieving information from the resource file in response to a request for a resource; providing the first worker with authority to modify the resource data-containing file and execute the application code; and restricting the first worker from accessing or modifying the application source code.